USING THE GNU RADIO PLATFORM TO IMPLEMENT A TELEMETRY RECEIVER

Gregory Newcomb, Ratish J. Punnoose

Sandia National Laboratories, P.O. Box 969, Livermore, CA 94551 USA

Email: rjpunno@sandia.gov

ABSTRACT

GNU Radio is a flexible software radio platform that enables custom radio development. It consists of open-source signal processing blocks that can be integrated into custom applications. The Universal Software Radio Peripheral (USRP) is a hardware board that works well with the GNU Radio suite. The schematics and firmware on this board are also open-source. As such, this GNU Radio and the USRP hardware form a rapid prototype platform for software radio based telemetry receivers.

I. INTRODUCTION

The GNU Radio software project is an open source collaborative effort to develop signal processing building blocks for radio applications [1]. These building blocks can facilitate the development of many media and software radio applications. The goal of the project is to implement software-based radio capabilities on personal computers using any available hardware. For example, the sound card on a desktop computer can be used to digitize some radio bands, which can then be processed in software. Similarly, HDTV and FM radio decoding using the GNU Radio platform have been demonstrated [2], [3]. It has also been used in amateur astronomy applications [4].

The Universal Software Radio Peripheral (USRP) is a hardware platform designed for use with the GNU Radio software [5]. The hardware consists of a Field Programmable Gate Array (FPGA), with Analog-to-Digital Converters (ADC), Digital-to-Analog Converters (DAC), and RF front-ends for different frequency ranges. Though this is a semi-commercial product, the schematics for the hardware and the FPGA firmware are available as open-source. The platform is designed so that its function can be modified or customized by the end-user.

We present the architecture of this combined open-source platform and describe the modifications required to implement a telemetry receiver.

II. The GNU Radio Project

The GNU Radio software is a freely available open-source signal processing toolkit aimed at RF applications. It is almost ten years old. It has its origins in software developed at MIT for the SpectrumWare project. The SpectrumWare project applies a software oriented approach for implementing radio communication technology. The grand goal was to use a general purpose computer for signal processing and interface it to an antenna through an ADC or DAC converter. The efforts of the SpectrumWare project have brought software radio closer to reality for many radio applications. One commercial success is the all software-radio based cellular phone base station by Vanu Inc.

The GNU Radio software, by extension of its past, aims at accomplishing all the signal processing routines on a general purpose processor, using portable, high-level software written in C++, and Python. It runs on Windows and on Linux based systems. It is not tied to any specific hardware for the analog to digital interface. The software is structured as a data flow diagram with modules that can input some data, process it, and pass it on to the next module. This is very much in line with other signal processing tools such as Labview or Simulink. For efficiency, the C++ language is used to implement the processing details within a module. The modules themselves can be organized and connected using Python, a high level scripting language. This allows for rapid implementation of a system completely in Python, which can manage the interconnections between the core signal processing blocks. As an addition, Python based GUI toolkits can be used to provide visualization and a graphical interface.

The set of included software modules is fairly comprehensive and is a good way to get started.

1) Basic math functions required for signal Processing, IIR, FIR, interpolating, and re-sampling filters.
2) Interleavers and de-interleavers, differential encoders, correlators.
3) PSK and FSK modulators and demodulators, trellis decoders and Viterbi modules, radar blocks.
4) Carrier Tracking, Clock recovery, PLL, AGC.
5) Audio and video codecs.
6) Error correction coding and decoding.

The software also supports several modules to interface with hardware to transmit and receive real signals.

1) USRP (Universal Software Radio Peripheral).
2) Sound cards
3) Video cards for output.
4) ATSC (HDTV) transmitter and receiver
5) Linux based data acquisition systems.

The most important thing to note is that GNU Radio provides a platform where these signal

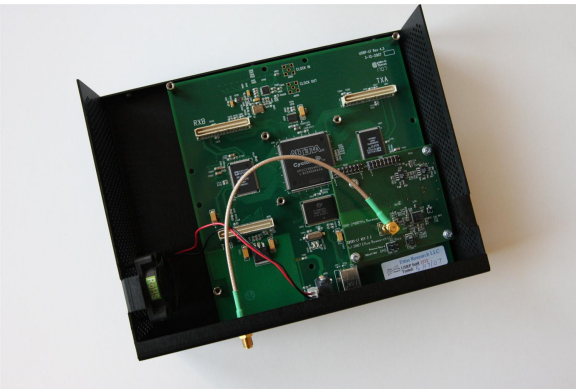

Fig. 1. A USRP board with an RF daughterboard.

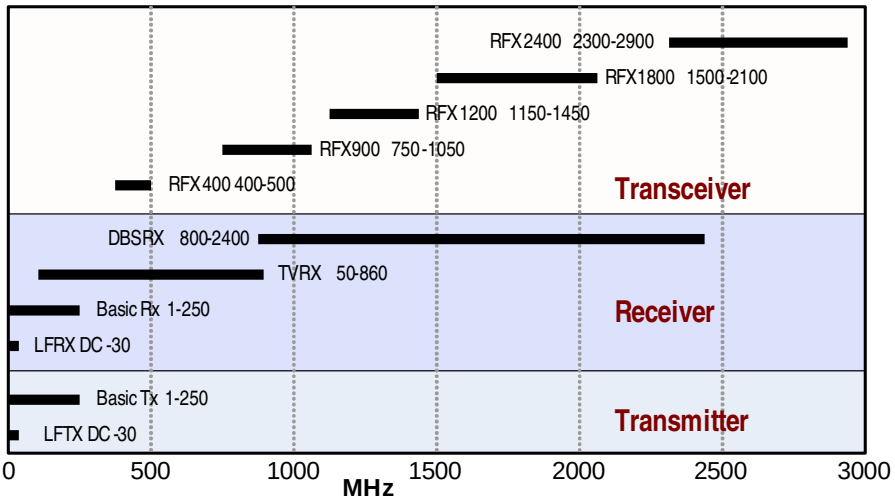

Fig. 2. USRP Daughterboard Options

processing blocks can be combined to build applications. Since it is open-source, custom blocks can be created by modifying existing blocks or from scratch.

III. The Universal Software Radio Peripheral

When the GNU Radio project got started, the only affordable A/D, D/A interfaces available to hobbyists were sound cards. The Universal Software Radio Peripheral (USRP) was built by Ettus Research to provide a radio-frequency A/D, D/A for use with the GNU Radio software. The core USRP motherboard consists of only high-speed A/Ds and D/As interfaced to an FPGA. A USB connection is provided to connect the board to a desktop computer. Daughter-boards supporting different analog interfaces can be stacked on to the motherboard (as much as four simultaneously). A USRP board with an RF daughterboard is shown in Figure 1. Several daughter cards for the USRP are available, supporting many commonly used RF frequency bands (Figure 2). The workhorse of the USRP motherboard is an Altera Cyclone FPGA. The FPGA interfaces to the daughterboards through a pair of Analog Devices Mixed Signal Processors that can perform

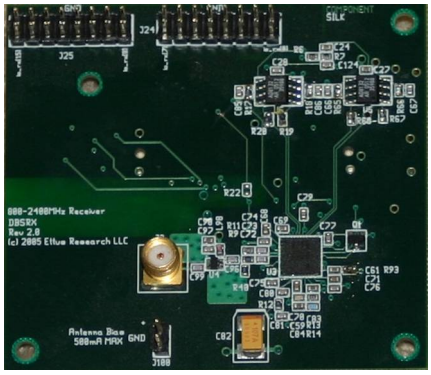

Fig. 3. DBSRX USRP RF daughter board

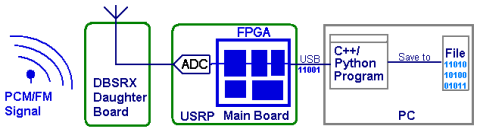

Fig. 4. Block diagram of our implementation of PCM/FM on this platform

A/D and D/A conversion. They can also be configured to perform some gain amplification, decimation, and interpolation. They support a maximum A/D sample rate of 64 Msps with 12 bits of resolution and a maximum D/A rate of 128 Msps with 14 bits of resolution. A dedicated USB chip provides a USB 2.0 interface to the board. Though, the USB interface can theoretically support 480 Mbps, in practice transfer rates are restricted by other factors rate limitations on the desktop computer's USB controller and processing required on the FPGA. Another feature of the hardware platform (motherboard and daughterboard) is that it supports synchronous operation. This is useful for Multiple-Input Multiple-Output (MIMO) applications where the phase differences between different RF streams have to be controlled.

IV. An Example Implementation of a Trivial FSK Demodulator on the GNU Radio Platform

To test the capabilities of the GNU Radio platform, we implemented a simple limiter-discriminator based IRIG Tier-0 receiver. We used the DBSRX daughter card as shown in Figure 4. For an IRIG telemetry operation in the 2.2 to 2.3 GHz band, the DBSRX daughterboard that provides reception in the 800 MHz to 2.4 GHz range, is eminently suitable (Figure 3). This daughterboard has a channel filter which is software programmable within a range of 1 to 60 MHz. The GNU Radio/USRP platform is structured so that most of the computation is done on the desktop computer and only basic decimation/interpolation is done on the hardware FPGA. Indeed, this is reflected in the choice of the FPGA. The Altera Cyclone FPGA excels at low power operation and is inexpensive, but its performance is low compared to other available options. The default firmware on the FPGA implements AGC scaling, decimation, filtering, and buffering for the

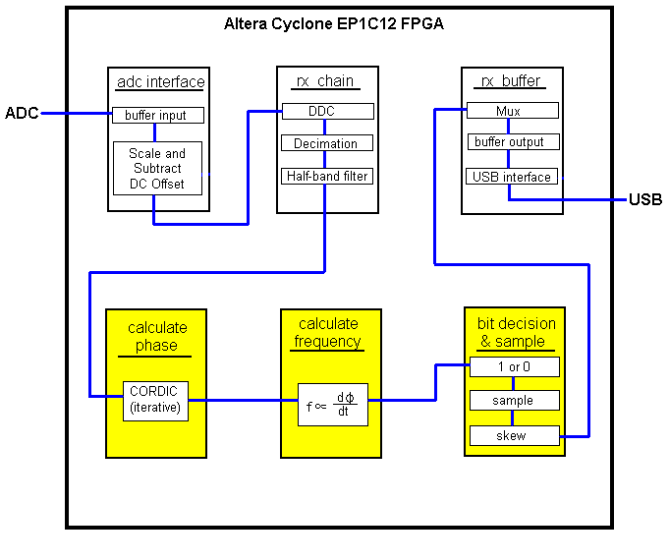

Fig. 5. Modifications in the FPGA firmware to implement the PCM/FM receiver

USB interface. Since these operations are not too complex, they fit adequately on the FPGA with some room to spare.

Our implementation needed demodulation and decommutation. We chose to implement the demodulation fully within the FPGA on the USRP. Most GNU Radio applications tend to do all complex signal processing on the host processor. Thus, our approach is unconventional approach compared to this philosophy. We chose this for several reasons: a) to test the ease with which we could modify the firmware on the USRP, b) to reduce the data flow over the USB interface (the USB controllers on some desktops cannot easily keep up without losing data), c) balancing the processing, since the host computer had to perform decommutation on the demodulated data. Though the FPGA on the USRP is limited, our choice of a simple limiter-discriminator approach was well within the spare capacity of the FPGA. Figure 5 shows the modified data flow on the USRP hardware. The three blocks on the top part of the diagram (adc_interface, rx_chain, rx_buffer) are connected to each other in the original implementation. We modified the flow between the rx_chain and the rx_buffer to pass through three additional blocks: a CORDIC calculator to compute phase, a frequency estimator, and a bit sampler. The CORDIC calculator is the more intensive of the three blocks [6], [7]. It uses the COordinate Rotation DIgital Computer (CORDIC) algorithm to compute the phase of the signal. The CORDIC algorithm iteratively rotates the phase of a complex number by multiplying it by a succession of constant values. However, since the multiplies are always powers of two, this can be done using shifts and adds. This is very efficient on an FPGA. This is done iteratively to decrease error (Figure 6). The algorithm works very naturally at a bit level and is parallelizable. This is a natural fit for an FPGA where parallelism is advantageous. We implemented a parallel version of the CORDIC algorithm to compute phase. The phase information was then used to compute frequency. The frequency signal was sliced to obtain the demodulated bits. Figure 7 shows the probed signals

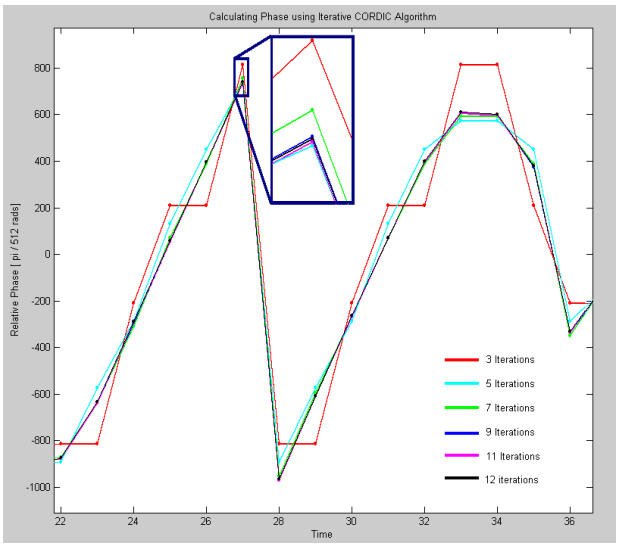

Fig. 6. Error in the CORDIC algorithm can be decreased by increasing iterations

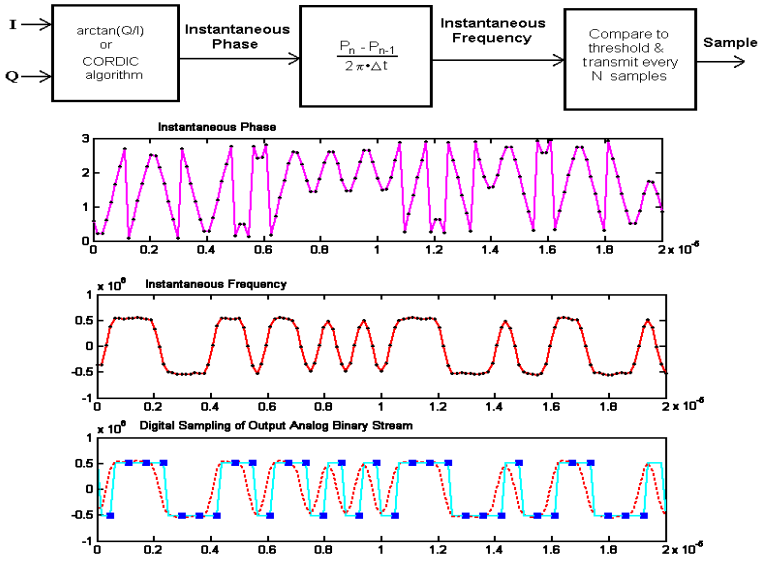

Fig. 7. A probe of the intermediate signals within the FPGA

in the FPGA in the process of demodulation. The demodulated data was then passed over the USB interface to the host computer. Using the GNU Radio blocks, the streaming data can be saved to file or it can be decommutated using custom blocks.

V. Conclusion

The GNU Radio software and the Universal Software Radio Peripheral provide a low-cost platform for software-radio development. The modular nature of the software allows the experimentalist to build system applications by connecting the signal processing blocks in unique

ways. The open-source collaborative nature of the project helps in the development of custom blocks. The platform offers development at three levels:

1) Script based development using Python, a high-level language, using the existing signal processing blocks. This approach is very fast and requires minimal effort. Graphical interfaces and visualization can also be done using the Python GUI toolkits.
2) Custom GNU Radio signal processing development in C++, and integration using Python. This requires some more work and an understanding of how to write the processing blocks. This is extremely flexible.
3) FPGA based development in Verilog on the USRP. The data can then be processed using existing or custom C++ blocks. The blocks can be connected using Python. This requires more work but is computationally efficient as the raw data can be greatly reduced before host processing.

In practice, a complex application may require some development at all of these levels. The only disadvantage of this platform is that documentation is not comprehensive and there is a learning curve (specially, if custom blocks have to be built).

We have used this platform to build a simple S-band telemetry receiver. Rather than using the existing signal-processing blocks, we implemented custom processing on the USRP hardware for efficiency. This provided a good test of the platform, since this allowed us to test modifications to both the USRP hardware and the GNU Radio software.

The GNU Radio software and the USRP firmware is constantly updated to provide new blocks and features. The next release of the USRP hardware aims to integrate a more powerful FPGA on-board and a higher speed interface (e.g. Gigabit Ethernet). The GNU Radio and the USRP will be good foundation for rapid deployment of the ever increasing number of wireless applications.

REFERENCES

[1] E. Blossom, "GNU Radio: Tools for Exploring the Radio Frequency Spectrum," *Linux Journal*, Jun 2004. http://www.linuxjournal.com/article/7319.

[2] E. Blossom, "GNU radio - HDTV snapshots." http://www.gnu.org/software/gnuradio/hdtv-samples.html.

[3] E. Blossom, "Listening to FM Radio in Software, Step by Step." http://www.linuxjournal.com/article/7505, Sep 2004.

[4] M. Leech, "GnuRadio and USRP: Solderless Breadboarding for the 21st Century," in *Society of Amateur Radio Astronomers*, (National Radio Astronomy Observatory, Green Banak WV), Jun 2006.

[5] http://www.ettus.com/.

[6] J. E. Volder, "The Birth of Cordic," *The Journal of VLSI Signal Processing*, vol. 25, pp. 101–105, Jun 2000.

[7] J. E. Volder, "The CORDIC Trigonometric Computing Techinque," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, 1959.